# **Project 1 .** Explore the Beauty of Automata

**See Section 3 on the deliverable and Section 4 for the due dates.**

## 1 Introduction

The goal of this project is for you to appreciate the beauty of Automata in the following sense:

1. Automata and its computation are defined precisely and elegantly using set theory language and logic. In fact, the definitions are pretty short and simple (mathematically).
2. The automata is also simple in the sense that you can implement a automata in a rather short time using a high level language. Imagine how hard it may be for you to implement a real CPU even using a high level language.
3. Although automata are simple, but some of them (e.g., Turing machine) are even more powerful than any existing CPU!
4. The simplicity of automata allows Cook (a computer scientist and Turing award winner) to study the hardness of computational problems! He finally discovered the concept of *NP-complete problems* for which it is very unlikely we can find efficient algorithms, i.e., algorithms that can solve a problem in polynomial time of the size of the problem.

We should also realize that the goal of automata (at least for this course) is to capture the substance of what computation is but NOT how to solve problems efficiently. For example, it may take us much longer time to construct an automata than to write a program using a high level programming language. Important goals of programming languages are to make programmers more effective and to help them write correct programs, but they are not the goals of automata.

## 2 Task

**Your Task**. write a program using Python, C/C++ or a language you negotiate with TA with the following functionality:

1. read from a file with the **format** consisting of a tuple $(\alpha, \beta)$ where $\alpha$ is a tuple for an NFA as explained in Section 2.1 below, and $\beta$ is a tuple $(t_1, \ldots, t_n)$ where for $i \in 1..n$, $t_i$ is a string over the alphabet of the NFA.(See the provided proj-1-machine.txt and proj-1-machine-1.txt.)

2. if $\beta$ is empty tuple $()$, your program will ask user to type from the keyboard an input string and display whether the input string is accepted or rejected, and repeat this process until the user give an empty input string (as shown in the execution example below). If $\beta$ is not empty, your program display a tuple of format $(r_1, \ldots, r_n)$ where for $i \in 1..n$, $r_i$ is *accepted* if the string $t_i$ is accepted by the NFA, or *rejected* otherwise. For example, for an input file consisting of $(\alpha, (1101, 0001, 1110))$ where $\alpha$ is the tuple representation of the NFA in Figure 1, your program will display the following and quit:

   ```
   (accepted, accepted, rejected)
   ```

3. Assumptions on the input file:

   - The maximum number of states allowed is 100 and the maximum number of symbols of the alphabet allowed is 50.
   - All the strings in $\beta$ of an input file will contain only symbols from the alphabet $\Sigma$.

Assume your executable program has name `runNFA`. An execution looks like the following:

```
> runNFA
Please input the file name:  proj-1-machine-1
Please input a string: 1001
Accepted.
Please input another string: 11111
Rejected.
Please input another string:
Bye bye.
```

### 2.1 Tuple representing an NFA

For any NFA $(\Sigma, K, s, F, \Delta)$, each state of $K$ is represented by an identifier which starts with a letter and then has any number of letters or numbers. In the input file to your program, the NFA is represented by the tuple $(\Sigma', K', s', F', \Delta')$ as follows

1. $\Sigma'$ is a tuple consisting of the symbols of $\Sigma$,
2. $K'$ a tuple of all states of $K$,
3. $s'$ the start state $s$,
4. $F'$ a tuple consisting of the final states of $F$,
5. $\Delta'$ a tuple representing the transition relation $\Delta$ and each element of $\Delta'$ is a triple $(s, c, t) \in \Delta$.

Consider an example NFA:

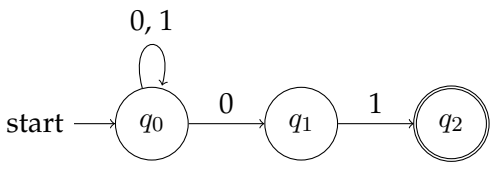


Figure 1: Example NFA

The tuple representation of the example NFA is $(\Sigma', K', s', F', \Delta')$ where

$\Sigma' = (0, 1)$

$K' = (q_0, q_1, q_2)$

$s' = q_0$

$F' = (q_2)$

$\Delta' = ((q_0, 0, q_0), (q_0, 1, q_0), (q_0, 0, q_1), (q_1, 1, q_2))$.

In another form, the NFA is

( ( 0, 1),
  ( $q_0, q_1, q_2$),
  $q_0$,
  ( $q_2$),
  ( ($q_0, 0, q_0$),
    ($q_0, 1, q_0$),
    ($q_0, 0, q_1$),
    ($q_1, 1, q_2$)
  )
)

# 3 Requirements and Deliverables

1. **Credit.** The project will be 16% of your final score of this course (the *assignment* portion on the syllabus).

2. **Team.** This is a team project and you must form a team of 2 - 3 member(s).

3. **Option:** One can choose to allow only **deterministic finite automaton (DFA)** instead of NFA in this project. In the case, maximal grade will be **85% of the total grade**. For those who do not think the time spent on NFA is worthwhile but still want to get a decent grade and understanding of the goals of this project, this option may be a good one.

4. **What to submit:** Submit to Canvas a single zip file, by the whole group, containing the report that must be in PDF format, the source code and readme.txt.

   (a) Include all the names of all team members in every file: the report, the source code and readme.txt.

   (b) Report - below are sons rough guidelines

      i. At the beginning of your report you must include one of the following: we implement DFA or we implement NFA.

      ii. Introduction section: include anything that you would like to attract our attention.

      iii. Specify the important data structure(s). The specification should be independent of any specific programming language.

iv. The pseudo-code of the algorithm. (See the template report - `https://docs.google.com/document/d/1KdmeCqchlkkB1U6s6yk_keZeZ6Iqt-wS/edit?usp=sharing&ouid=114871752263027921289&rtpof=true&sd=true`)

v. The test cases to test the correctness of your program with explanation why you select them. Each case should be of the format as defined in Section 2.

vi. Acknowledgment of people and their contribution to your project.

(c) source codes: use proper indentations and comments.

(d) a readme.txt file containing all the info about software/applications, programming language used to complete the project,and how to compile and run your program in command line under Windows or Mac.

# 4 Important Dates and Information

- **Team:** It is your responsibility to form your team and **send your team members to our TA** by **11:59pm Nov 6**. Please write to our TA, by **11:59pm Nov 6** if you can not set up a team, and he will surely help you.
- **Project Due Date: 11:59pm Nov 23**. Only one member needs to submit all required materials. You have to designate a second member to make sure the other member submit the materials ON TIME.
- **Interview Dates: Nov 24, 25**. One will get 0 if they are absent from the interview. There will be 15 min interview for each group. Every student is expected to know well everything in the report or source code. No matter how well the source code and report are, a student can get very bad score if they can not answer well questions about the project (report or source code). The rule of thumb is that every member should know why **every word / piece of code** in the report or source code is used.

# 5 About the algorithm for simulating the NFA

During the computation deciding whether a given string is accepted by the automaton, there may be several states to go next. However, every time we can only go to one of them. You can randomly select one, while saving all other states, together with the rest of the input string, in a stack so that we can come back to explore them later when we get stuck. It is good idea for you to elaborate this idea using a very small example in your report. You also want to make this idea clear and explicit in your pseudo-code of the algorithm. More details can be found from the file `proj-1-algorithmIdeas.pdf`.

## 6 Grading

| **Rubric** | |
|---|---|
| Points | Criteria |
| 25 | clear and complete report submission |
| 15 | correct, clear pseudo-code of the algorithm |
| 20 | program is successful for the test cases |
| 20 | correct implementation |
| 5 | inline comments to briefly describe your code |
| 15 | implementation and algorithm in report addressing NFA |
| | **Interview results** can possibly affect all your points above |